

# SERVERLESS

## Проблемы разработки

---



# ПЛАН

---

1. Мир Serverless
2. Особенности реализации
3. Выводы





# О СЕБЕ

Руслан Серкин

Software engineer  
DataArt

Email: [ruslan.ru@gmail.com](mailto:ruslan.ru@gmail.com)

Twitter: [@ruslan\\_firefy](https://twitter.com/ruslan_firefy)



# ПРОЕКТ

---

- Автоматизация оффлайн бизнеса



# ПРОЕКТ

---

- Автоматизация оффлайн бизнеса
- Команда: JavaScript разработчики



# ПРОЕКТ

---

- Автоматизация оффлайн бизнеса
- Команда: JavaScript разработчики
- AWS



---

# SERVERLESS

---



# SERVERLESS - ПРИМЕНЕНИЕ

---

- Sites / API



# SERVERLESS - ПРИМЕНЕНИЕ

---

- Sites / API
- Internet of things



# SERVERLESS - ПРИМЕНЕНИЕ

---

- Sites / API
- Internet of things
- Периодические задания



# SERVERLESS - ПРИМЕНЕНИЕ

---

- Sites / API
- Internet of things
- Периодические задания
- Рассылка писем



# SERVERLESS - ПРИМЕНЕНИЕ

---

- Sites / API
- Internet of things
- Периодические задания
- Рассылка писем
- Сервисные задания



# SERVERLESS - ПРИМЕНЕНИЕ

---

- Sites / API
- Internet of things
- Периодические задания
- Рассылка писем
- Сервисные задания
- Другое



# SERVERLESS - ПРОВАЙДЕРЫ

---

- AWS Lambda (AWS)



# SERVERLESS - ПРОВАЙДЕРЫ

---

- AWS Lambda (AWS)
- Azure Functions (Azure)



# SERVERLESS - ПРОВАЙДЕРЫ

---

- AWS Lambda (AWS)
- Azure Functions (Azure)
- Cloud Functions (GCP)



# SERVERLESS - ПРОВАЙДЕРЫ

---

- AWS Lambda (AWS)
- Azure Functions (Azure)
- Cloud Functions (GCP)
- Kubeless (K8s)



# SERVERLESS - ПРОВАЙДЕРЫ

---

- AWS Lambda (AWS)
- Azure Functions (Azure)
- Cloud Functions (GCP)
- Kubeless (K8s)
- Другие решения



# SERVERLESS - ПРЕИМУЩЕСТВА

---

- Просто функция



# SERVERLESS - ПРИМЕР (AWS)

---

```
"use strict";
module.exports.endpoint = (event, context, callback) => {
  const response = {
    statusCode: 200,
    body: JSON.stringify({
      message: `Hello, the current time is
        ${new Date().toLocaleTimeString()}`
    })
  };

  callback(null, response);
};
```



# SERVERLESS - ПРЕИМУЩЕСТВА

---

- Просто функция
  - Легко тестировать



# SERVERLESS - ПРЕИМУЩЕСТВА

---

- Просто функция
  - Легко тестировать
  - Меньше зависимостей



# SERVERLESS - ПРЕИМУЩЕСТВА

---

- Просто функция
  - Легко тестировать
  - Меньше зависимостей
  - Идеальный микросервис



# SERVERLESS - ПРЕИМУЩЕСТВА

---

- Деплой



# SERVERLESS - ПРЕИМУЩЕСТВА

---

- Деплой
  - Zip архив



# SERVERLESS - ПРЕИМУЩЕСТВА

---

- Деплой
  - Zip архив
  - Интерфейс



# SERVERLESS - ПРЕИМУЩЕСТВА

---

- Деплой
  - Zip архив
  - Интерфейс
  - Google repos



# SERVERLESS - ПРЕИМУЩЕСТВА

---

- Деплой
  - Zip архив
  - Интерфейс
  - Google repos
  - Утилиты и плагины для IDE



# SERVERLESS - AWS CLOUD9

devoops-test1-dev-hello

Throttle

Qualifiers ▼

Actions ▼

Code entry type

Edit code inline ▼

Runtime

Node.js 8.10 ▼

Handler

handle

File Edit Find View Goto Tools Window

Environment

devoops-test1-dev-hello

deploy.sh

handler.js

handler.js

handler.js

```
1 |'use strict';
2 module.exports.hello = (event, context, callback) => {
3   const response = {
4     statusCode: 200,
5     body: JSON.stringify({
6       message: `Hello, the current time is ${new Date().toLocaleTimeString()}`
7     })
8   };
9   callback(null, response);
10 };
```



# SERVERLESS - ПРЕИМУЩЕСТВА

---

- Масштабируемость



# SERVERLESS - ПРЕИМУЩЕСТВА

---

- Масштабируемость
  - Нет серверов



# SERVERLESS - ПРЕИМУЩЕСТВА

---

- Масштабируемость
  - Нет серверов
  - Функция на событие (API Gateway, SNS, SQS...)



# SERVERLESS - ПРЕИМУЩЕСТВА

---

- Масштабируемость
  - Нет серверов
  - Функция на событие (API Gateway, SNS, SQS...)
  - Быстрый старт



# SERVERLESS - ПРЕИМУЩЕСТВА

---

- Цена



# SERVERLESS - ПРЕИМУЩЕСТВА

---

- Цена
  - Нет месячной подписки



# SERVERLESS - ПРЕИМУЩЕСТВА

---

- Цена
  - Нет месячной подписки
  - Платим за время и ресурсы



# SERVERLESS - ПРЕИМУЩЕСТВА

---

- Цена
  - Нет месячной подписки
  - Платим за время и ресурсы
  - [serverlesscalc.com](https://serverlesscalc.com)



# SERVERLESS - ПРЕИМУЩЕСТВА

---

- Языки программирования



# SERVERLESS - ПРЕИМУЩЕСТВА

---

- Языки программирования
  - Python



# SERVERLESS - ПРЕИМУЩЕСТВА

---

- Языки программирования
  - Python
  - NodeJS



# SERVERLESS - ПРЕИМУЩЕСТВА

---

- Языки программирования
  - Python
  - NodeJS
  - Go



# SERVERLESS - ПРЕИМУЩЕСТВА

---

- Языки программирования
  - Python
  - NodeJS
  - Go
  - Java



# SERVERLESS - ПРЕИМУЩЕСТВА

---

- Языки программирования
  - Python
  - NodeJS
  - Go
  - Java
  - C#



# СРАВНЕНИЕ ЯЗЫКОВ НА AWS

	Типы				
<b>GO</b>	✓✓				
<b>NodeJS</b>	✗				
<b>Python</b>	✗				
<b>Java</b>	✓✓				
<b>C#</b>	✓✓				



# СРАВНЕНИЕ ЯЗЫКОВ НА AWS

	Типы	Деплой			
<b>GO</b>	✓✓	✓			
<b>NodeJS</b>	✗	✓✓			
<b>Python</b>	✗	✓✓			
<b>Java</b>	✓✓	✗			
<b>C#</b>	✓✓	✓			



# СРАВНЕНИЕ ЯЗЫКОВ НА AWS

	Типы	Деплой	Старт		
GO	✓✓	✓	✓		
NodeJS	✗	✓✓	✓✓		
Python	✗	✓✓	✓✓		
Java	✓✓	✗	✗		
C#	✓✓	✓	✗		



# СРАВНЕНИЕ ЯЗЫКОВ НА AWS

	Типы	Деплой	Старт	Скорость
<b>GO</b>	✓✓	✓	✓	✓✓
<b>NodeJS</b>	✗	✓✓	✓✓	✓
<b>Python</b>	✗	✓✓	✓✓	✓
<b>Java</b>	✓✓	✗	✗	✓✓
<b>C#</b>	✓✓	✓	✗	✓✓



# СРАВНЕНИЕ ЯЗЫКОВ НА AWS

	Типы	Деплой	Старт	Скорость	Экосистема
<b>GO</b>	✓✓	✓	✓	✓✓	✓✓
<b>NodeJS</b>	✗	✓✓	✓✓	✓	✓
<b>Python</b>	✗	✓✓	✓✓	✓	✓✓
<b>Java</b>	✓✓	✗	✗	✓✓	✗
<b>C#</b>	✓✓	✓	✗	✓✓	✗



# СРАВНЕНИЕ ПЛАТФОРМ

	<b>AWS</b>	<b>Azure</b>	<b>GCP</b>	<b>Kubeless</b>
<b>Деплой</b>	ZIP / Web Editor	ZIP / Web Editor	ZIP/ Google repo	CLI



# СРАВНЕНИЕ ПЛАТФОРМ

	<b>AWS</b>	<b>Azure</b>	<b>GCP</b>	<b>Kubeless</b>
<b>Деплой</b>	ZIP / Web Editor	ZIP / Web Editor	ZIP/ Google repo	CLI
<b>Языки</b>	Java, JS, C#, Go, Python	C#, F#, JS, Python, Java, PHP	JavaScript	Python, JS, Go, Ruby, PHP, C#



# СРАВНЕНИЕ ПЛАТФОРМ

	<b>AWS</b>	<b>Azure</b>	<b>GCP</b>	<b>Kubeless</b>
<b>Деплой</b>	ZIP / Web Editor	ZIP / Web Editor	ZIP/ Google repo	CLI
<b>Языки</b>	Java, JS, C#, Go, Python	C#, F#, JS, Python, Java, PHP	JavaScript	Python, JS, Go, Ruby, PHP, C#
<b>Зависимости</b>	стандартные пакеты языка	стандартные пакеты языка	стандартные пакеты языка	стандартные пакеты языка



# СРАВНЕНИЕ ПЛАТФОРМ

	<b>AWS</b>	<b>Azure</b>	<b>GCP</b>	<b>Kubeless</b>
<b>Деплой</b>	ZIP / Web Editor	ZIP / Web Editor	ZIP/ Google repo	CLI
<b>Языки</b>	Java, JS, C#, Go, Python	C#, F#, JS, Python, Java, PHP	JavaScript	Python, JS, Go, Ruby, PHP, C#
<b>Зависимости</b>	стандартные пакеты языка	стандартные пакеты языка	стандартные пакеты языка	стандартные пакеты языка
<b>Триггеры</b>	AWS Services	AZURE Services	Cloud pub/sub, Storage, HTTP	Kafka, NATS, HTTP



# СРАВНЕНИЕ ПЛАТФОРМ

	<b>AWS</b>	<b>Azure</b>	<b>GCP</b>	<b>Kubeless</b>
<b>Деплой</b>	ZIP / Web Editor	ZIP / Web Editor	ZIP/ Google repo	CLI
<b>Языки</b>	Java, JS, C#, Go, Python	C#, F#, JS, Python, Java, PHP	JavaScript	Python, JS, Go, Ruby, PHP, C#
<b>Зависимости</b>	стандартные пакеты языка	стандартные пакеты языка	стандартные пакеты языка	стандартные пакеты языка
<b>Триггеры</b>	AWS Services	AZURE Services	Cloud pub/sub, Storage, HTTP	Kafka, NATS, HTTP
<b>Мониторинг</b>	CloudWatch, X-Ray	Application Insights	StackDriver	Prometheus



# ПРОЕКТ

---

- Мобильные приложения



# ПРОЕКТ

---

- Мобильные приложения
- Сайт



# ПРОЕКТ

---

- Мобильные приложения
- Сайт
- Точки продаж с системой распознавания QR



# ПРОЕКТ

---

- Мобильные приложения
- Сайт
- Точки продаж с системой распознавания QR
- Несколько регионов



# КОНФИГУРАЦИЯ

---

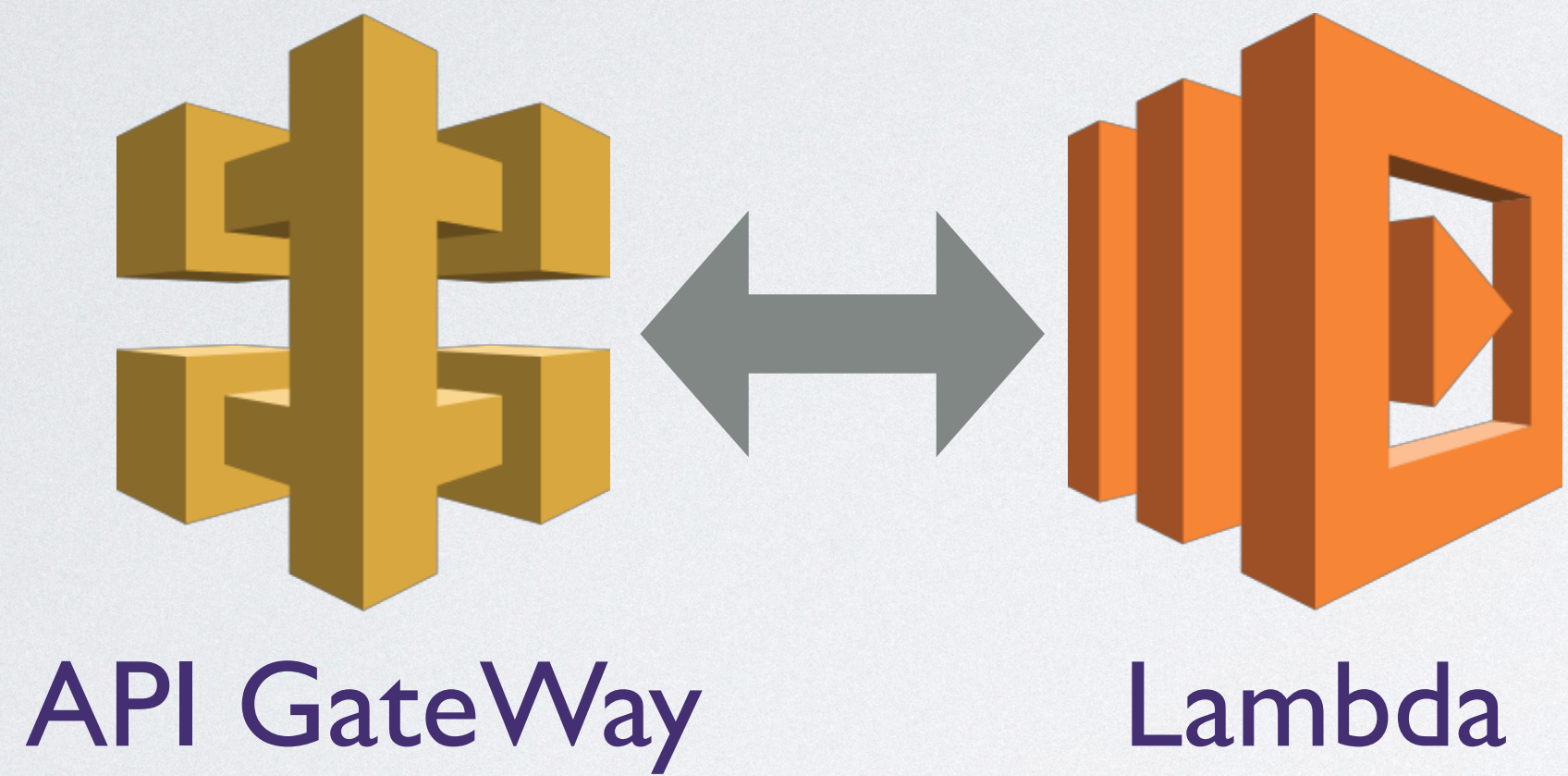


API GateWay



# КОНФИГУРАЦИЯ

---





# КОНФИГУРАЦИЯ

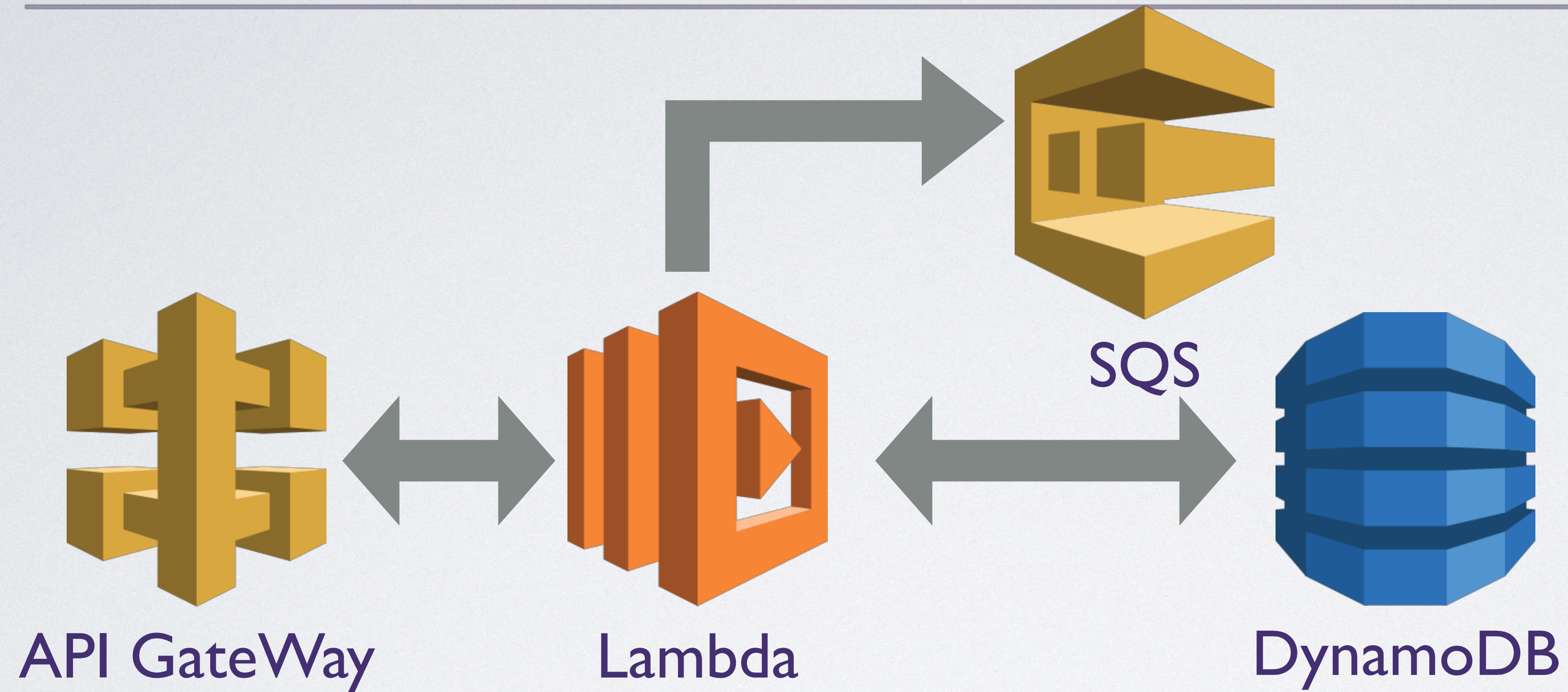
---





# КОНФИГУРАЦИЯ

---





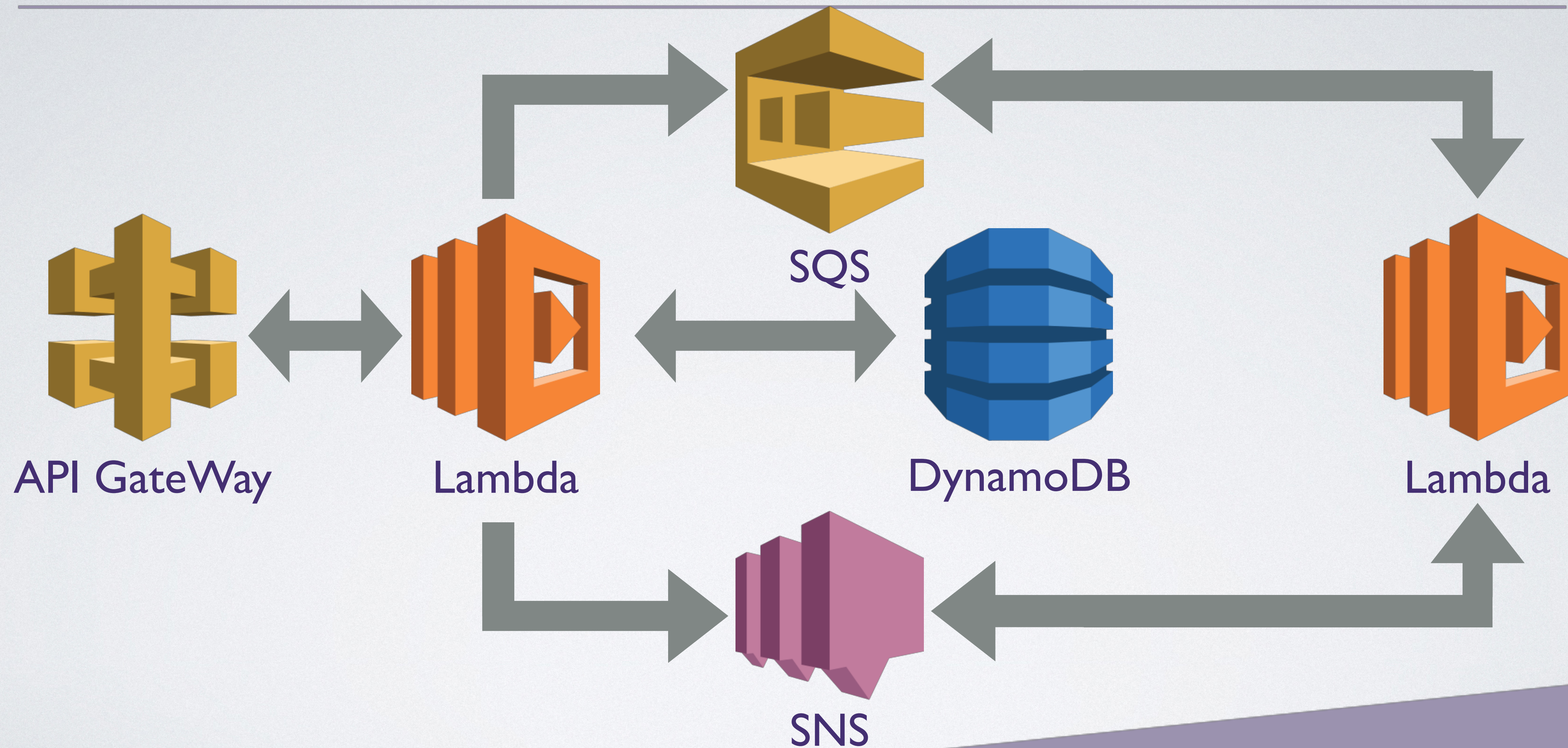
# КОНФИГУРАЦИЯ

---



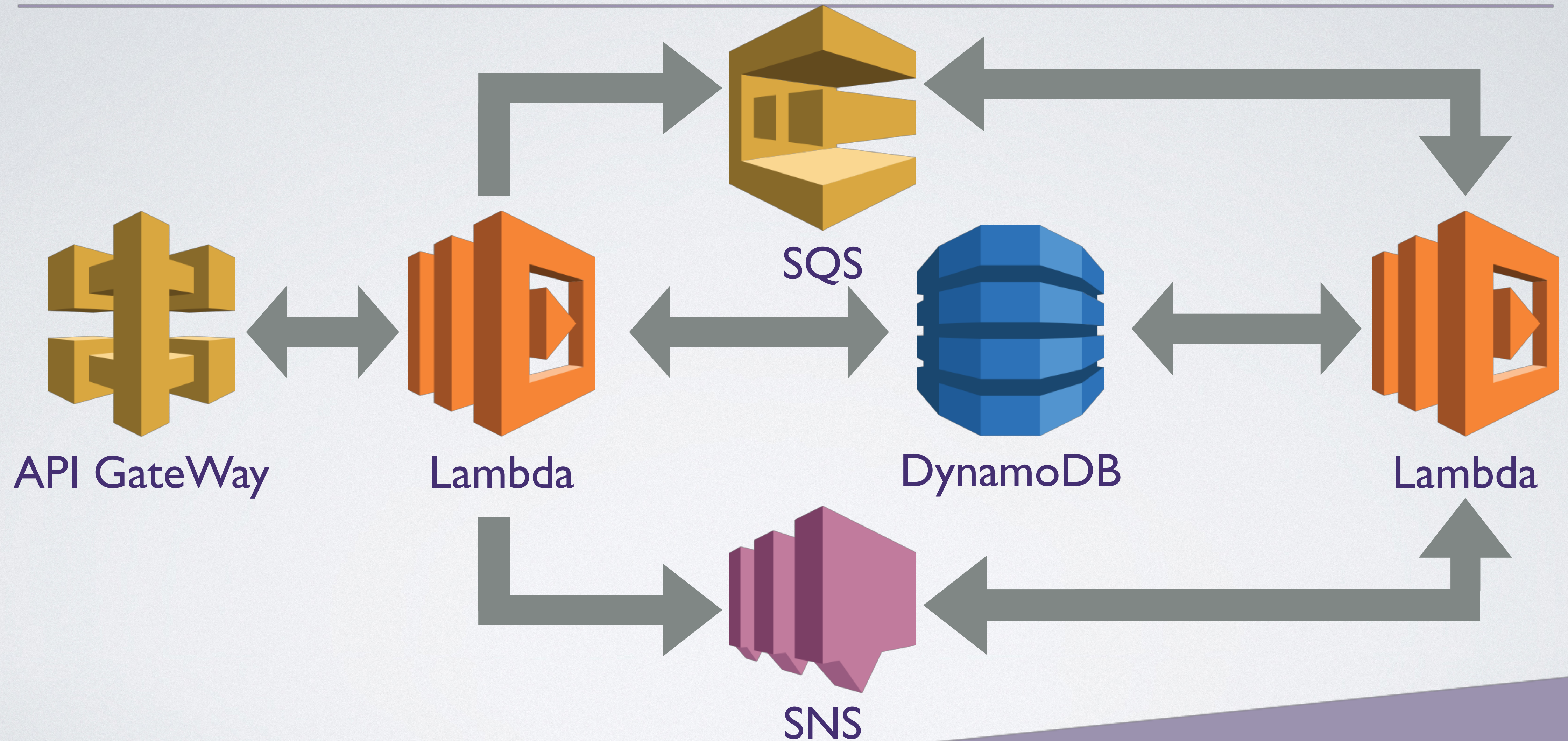


# КОНФИГУРАЦИЯ





# КОНФИГУРАЦИЯ





---

# ЛОГИРОВАНИЕ

---



# ЛОГИРОВАНИЕ

---

- AWS CloudWatch



# ЛОГИРОВАНИЕ

---

- AWS CloudWatch
- ELK



# ЛОГИРОВАНИЕ

---

- AWS CloudWatch
- ELK
- Внешний сервис



# ЛОГИРОВАНИЕ - CLOUDWATCH

---

- Простота интеграции



# ЛОГИРОВАНИЕ - CLOUDWATCH

---

- Простота интеграции
- Готовая инфраструктура



# ЛОГИРОВАНИЕ - CLOUDWATCH

---

- Простота интеграции
- Готовая инфраструктура
- Пишем в `stdout`



# ЛОГИРОВАНИЕ - CLOUDWATCH

```
handler.js x (+)
1 "use strict";
2 module.exports.hello = (event, context, callback) => {
3
4   console.log("Hello devoops conf");
5
6   const response = {
7     statusCode: 200,
8     body: JSON.stringify({
9       message: `Hello, the current time is ${new Date().toLocaleTimeString()}`
10    })
11  };
12  callback(null, response);
13 }
```



# ЛОГИРОВАНИЕ - CLOUDWATCH

---

## Message

2018-09-28 18:23:39

*No older events found at t*

START RequestId: 9f4e43f7-c34b-11e8-ba6b-45e62ff1f254 Version: \$LATEST

2018-09-28T18:23:39.952Z 9f4e43f7-c34b-11e8-ba6b-45e62ff1f254 Hello devoops conf

END RequestId: 9f4e43f7-c34b-11e8-ba6b-45e62ff1f254

REPORT RequestId: 9f4e43f7-c34b-11e8-ba6b-45e62ff1f254 Duration: 3.03 ms Billed Durati



# ЛОГИРОВАНИЕ - CLOUDWATCH

---

- Неудобно читать логи



# ЛОГИРОВАНИЕ - CLOUDWATCH

---

- Неудобно читать логи
- Тяжело построить контекст



# ЛОГИРОВАНИЕ - CLOUDWATCH

---

- Неудобно читать логи
- Тяжело построить контекст
- X-Ray (<http://clc.to/JQi-zA>)



# ЛОГИРОВАНИЕ - CLOUDWATCH

---

- Неудобно читать логи
- Тяжело построить контекст
- X-Ray (<http://clc.to/JQi-zA>)
  - нет данных в реальном времени



# ЛОГИРОВАНИЕ - CLOUDWATCH

---

- Неудобно читать логи
- Тяжело построить контекст
- X-Ray (<http://clc.to/JQi-zA>)
  - нет данных в реальном времени
  - не всегда данные достоверны



# ЛОГИРОВАНИЕ - CLOUDWATCH

---

- Неудобно читать логи
- Тяжело построить контекст
- X-Ray (<http://clc.to/JQi-zA>)
  - нет данных в реальном времени
  - не всегда данные достоверны
- Логи сгруппированы



# ЛОГИРОВАНИЕ - CLOUDWATCH

Search Log Group   Create Log Stream   Delete Log Stream

Filter: Log Stream Name Prefix ×

<input type="checkbox"/>	Log Streams	Last Event Time
<input type="checkbox"/>	2018/09/28/[\$LATEST]1ec52a77fdf6473b8fe6d97847aea8e5	2018-09-28 21:25 UTC+3
<input type="checkbox"/>	2018/09/28/[\$LATEST]200db69fcd624b13b211ee3b16bc95cb	2018-09-28 21:21 UTC+3



# ЛОГИРОВАНИЕ - ELK / EXTERNAL

---

- Куда писать



# ЛОГИРОВАНИЕ - ELK / EXTERNAL

---

- Куда писать
  - AWS CloudWatch



# ЛОГИРОВАНИЕ - ELK / EXTERNAL

---

- Куда писать
  - AWS CloudWatch
  - Внешний сервис из кода



# ЛОГИРОВАНИЕ - ELK / EXTERNAL

---

- Куда писать
  - AWS CloudWatch
  - Внешний сервис из кода
  - Stdout



# ЛОГИРОВАНИЕ - ELK / EXTERNAL

---

- Куда писать
  - AWS CloudWatch
  - Внешний сервис из кода
  - Stdout
- Простой поиск



# ЛОГИРОВАНИЕ - ELK / EXTERNAL

---

- Куда писать
  - AWS CloudWatch
  - Внешний сервис из кода
  - Stdout
- Простой поиск
- Построение графиков



# ЛОГИРОВАНИЕ - TIMEOUTS

---

- Функция не работает бесконечно



# ЛОГИРОВАНИЕ - TIMEOUTS

---

- Функция не работает бесконечно
  - Максимально 300 секунд



# ЛОГИРОВАНИЕ - TIMEOUTS

---

- Функция не работает бесконечно
  - Максимально 300 секунд
- Потеря данных



# ЛОГИРОВАНИЕ - TIMEOUTS

---

- Функция не работает бесконечно
  - Максимально 300 секунд
- Потеря данных
  - Повторный запуск после ошибки



# ЛОГИРОВАНИЕ - TIMEOUTS

---

- Функция не работает бесконечно
  - Максимально 300 секунд
- Потеря данных
  - Повторный запуск после ошибки
  - Создавать свой таймер



---

# ДЕПЛОЙ И КОНФИГУРАЦИЯ

---



# КОНФИГУРАЦИЯ И ДЕПЛОЙ

---

- Интерфейс



# КОНФИГУРАЦИЯ И ДЕПЛОЙ

---

- Интерфейс
- CLI



# КОНФИГУРАЦИЯ И ДЕПЛОЙ

---

- Интерфейс
- CI
- Плагины для IDE



# КОНФИГУРАЦИЯ ЧЕРЕЗ ИНТЕРФЕЙС

---

- Редактор кода



# КОНФИГУРАЦИЯ ЧЕРЕЗ ИНТЕРФЕЙС

---

- Редактор кода
- Установка переменных окружения



# КОНФИГУРАЦИЯ ЧЕРЕЗ ИНТЕРФЕЙС

## Environment variables

You can define Environment Variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. [Learn more.](#)

NODE_VAR	TEST_VAR	Remove
<i>Key</i>	<i>Value</i>	Remove

### ▼ Encryption configuration

Enable helpers for encryption in transit [Info](#)

AWS KMS key to encrypt at rest [Info](#)

Choose an AWS KMS key to encrypt the environment variables at rest, or simply let Lambda manage the encryption.

- (default) aws/lambda
- Use a customer master key



# КОНФИГУРАЦИЯ ЧЕРЕЗ ИНТЕРФЕЙС

---

- Редактор кода
- Установка переменных окружения
- Удобно подписываться на события



# КОНФИГУРАЦИЯ ЧЕРЕЗ ИНТЕРФЕЙС

The screenshot shows the 'Add triggers' configuration page in the AWS Lambda console. On the left, a sidebar lists available triggers: API Gateway, AWS IoT, and Alexa Skills Kit. The main area features a key icon and a list of triggers. The 'AWS IoT' trigger is highlighted in blue and includes a 'Configuration required' message. The 'Amazon CloudWatch Logs' trigger is also visible. A dashed box at the bottom left contains the text 'Add triggers from the list on the left', and another dashed box at the bottom right contains the text 'Resources that the function's role has access to'.

**Add triggers**  
Choose a trigger from the list below to add it to your function.

- API Gateway
- AWS IoT
- Alexa Skills Kit

**devoops-test1-dev-hello**

- AWS IoT** Configuration required
- Amazon CloudWatch Logs

Add triggers from the list on the left

Resources that the function's role has access to



# КОНФИГУРАЦИЯ ЧЕРЕЗ ИНТЕРФЕЙС

---

- Нет скриптов для разворачивания окружения



# КОНФИГУРАЦИЯ ЧЕРЕЗ ИНТЕРФЕЙС

---

- Нет скриптов для разворачивания окружения
- Трудности отладки функций



# КОНФИГУРАЦИЯ ЧЕРЕЗ ИНТЕРФЕЙС

---

- Нет скриптов для разворачивания окружения
- Трудности отладки функций
- Проблемы в миграции



# КОНФИГУРАЦИЯ

---

- Инфраструктура (Terraform, Ansible, CloudFormation)



# КОНФИГУРАЦИЯ

---

- Инфраструктура (Terraform, Ansible, CloudFormation)
- AWS CLI



# КОНФИГУРАЦИЯ

---

- Инфраструктура (Terraform, Ansible, CloudFormation)
- AWS CLI
  - Скрипты для разворачивания функций



# КОНФИГУРАЦИЯ

---

- Инфраструктура (Terraform, Ansible, CloudFormation)
- AWS CLI
  - Скрипты для разворачивания функций
- Frameworks



# КОНФИГУРАЦИЯ

---

- Инфраструктура (Terraform, Ansible, CloudFormation)
- AWS CLI
  - Скрипты для разворачивания функций
- Frameworks
  - Унифицированный yaml конфиг



# КОНФИГУРАЦИЯ

---

- Инфраструктура (Terraform, Ansible, CloudFormation)
- AWS CLI
  - Скрипты для разворачивания функций
- Frameworks
  - Унифицированный yaml конфиг
  - Автоматическое создание API Gateway, SQS и тд



# FRAMEWORKS

---

- Apex (<http://apex.run/>)



# FRAMEWORKS

---

- Apex (<http://apex.run/>)
- Zappa (<https://www.zappa.io/>)



# FRAMEWORKS

---

- Apex (<http://apex.run/>)
- Zappa (<https://www.zappa.io/>)
- Sparta (<http://gosparta.io/>)



# FRAMEWORKS

---

- Apex (<http://apex.run/>)
- Zappa (<https://www.zappa.io/>)
- Sparta (<http://gosparta.io/>)
- Serverless (<https://serverless.com/>)



# SERVERLESS FRAMEWORK

---

- JavaScript



# SERVERLESS FRAMEWORK

---

- JavaScript
- Plugins



# SERVERLESS FRAMEWORK

---

- JavaScript
- Plugins
- Абстракция над облаками



# SERVERLESS FRAMEWORK - PLUGINS

---

- Запуск функции локально



# SERVERLESS FRAMEWORK - PLUGINS

---

- Запуск функции локально
- Эмуляция ApiGateway, SQS и так далее



# SERVERLESS FRAMEWORK - PLUGINS

---

- Запуск функции локально
- Эмуляция ApiGateway, SQS и так далее
- Логирование и мониторинг



# КОНФИГУРАЦИЯ

```
Serverless: Packaging service...
Serverless: Excluding development dependencies...
Serverless: Creating Stack...
Serverless: Checking Stack create progress...
CloudFormation - CREATE_IN_PROGRESS - AWS::CloudFormation::Stack - devoops-test1-dev
CloudFormation - CREATE_IN_PROGRESS - AWS::S3::Bucket - ServerlessDeploymentBucket
CloudFormation - CREATE_IN_PROGRESS - AWS::S3::Bucket - ServerlessDeploymentBucket
CloudFormation - CREATE_COMPLETE - AWS::S3::Bucket - ServerlessDeploymentBucket
CloudFormation - CREATE_COMPLETE - AWS::CloudFormation::Stack - devoops-test1-dev
Serverless: Stack create finished...
Serverless: Uploading CloudFormation file to S3...
Serverless: Uploading artifacts...
Serverless: Uploading service .zip file to S3 (576 B)...
Serverless: Validating template...
Serverless: Updating Stack...
Serverless: Checking Stack update progress...
CloudFormation - UPDATE_IN_PROGRESS - AWS::CloudFormation::Stack - devoops-test1-dev
CloudFormation - CREATE_IN_PROGRESS - AWS::IAM::Role - IamRoleLambdaExecution
CloudFormation - CREATE_IN_PROGRESS - AWS::IAM::Role - IamRoleLambdaExecution
CloudFormation - CREATE_IN_PROGRESS - AWS::Logs::LogGroup - HelloLogGroup
CloudFormation - CREATE_IN_PROGRESS - AWS::Logs::LogGroup - HelloLogGroup
CloudFormation - CREATE_COMPLETE - AWS::Logs::LogGroup - HelloLogGroup
CloudFormation - CREATE_COMPLETE - AWS::IAM::Role - IamRoleLambdaExecution
CloudFormation - CREATE_IN_PROGRESS - AWS::Lambda::Function - HelloLambdaFunction
CloudFormation - CREATE_IN_PROGRESS - AWS::Lambda::Function - HelloLambdaFunction
CloudFormation - CREATE_COMPLETE - AWS::Lambda::Function - HelloLambdaFunction
CloudFormation - CREATE_IN_PROGRESS - AWS::Lambda::Version - HelloLambdaVersionWPuMBrWEiE5lUkZJv3ujxFHBFVx7GyQtV3RXWmFi4o
CloudFormation - CREATE_IN_PROGRESS - AWS::Lambda::Version - HelloLambdaVersionWPuMBrWEiE5lUkZJv3ujxFHBFVx7GyQtV3RXWmFi4o
CloudFormation - CREATE_COMPLETE - AWS::Lambda::Version - HelloLambdaVersionWPuMBrWEiE5lUkZJv3ujxFHBFVx7GyQtV3RXWmFi4o
CloudFormation - UPDATE_COMPLETE_CLEANUP_IN_PROGRESS - AWS::CloudFormation::Stack - devoops-test1-dev
CloudFormation - UPDATE_COMPLETE - AWS::CloudFormation::Stack - devoops-test1-dev
Serverless: Stack update finished...
```



# CONTINUOUS DELIVERY

---

- Внешний сервис (CodeShip)



# CONTINUOUS DELIVERY

---

- Внешний сервис (CodeShip)
  - Сборка zip архива в докере



# CONTINUOUS DELIVERY

---

- Внешний сервис (CodeShip)
  - Сборка zip архива в докере
  - Загрузка на S3



# CONTINUOUS DELIVERY

---

- Внешний сервис (CodeShip)
  - Сборка zip архива в докере
  - Загрузка на S3
  - Автоматическое переключение в девелопе



# CONTINUOUS DELIVERY

---

- Внешний сервис (CodeShip)
  - Сборка zip архива в докере
  - Загрузка на S3
  - Автоматическое переключение в девелопе
  - Ручное переключение в продакшене



# CONTINUOUS DELIVERY

---

- Для одного региона: внешний CD



# CONTINUOUS DELIVERY

---

- Для одного региона: внешний CD
- Для мультирегиона: Jenkins, Teamcity



---

# DATABASE

---



# DATABASES

---

- DynamoDB



# DATABASES

---

- DynamoDB
- SQL



# SQL

---

При росте трафика мы получем ошибку при  
подключении к базе данных



# SQL

---

При росте трафика мы получем ошибку при  
подключении к базе данных

SQLSTATE[08004] [1040] Too many connections



# DATABASES

---

- Масштабируемость



# DATABASES

---

- Масштабируемость
  - Горячие контейнеры



# DATABASES

---

GET /users => getUsers()

PUT /user/:id => updateUser(id)

POST /user => saveUser()

DELETE /user/:id => deleteUser(id)



# DATABASES

---

GET /users => getUsers() 30/30

PUT /user/:id => updateUser(id) 30/30

POST /user => saveUser() 30/30

DELETE /user/:id => deleteUser(id) 30/30

Итого: 120



# DATABASES

---

GET /users => getUsers() 25/30

PUT /user/:id => updateUser(id) 10/30

POST /user => saveUser() 50/50

DELETE /user/:id => deleteUser(id) 20/30

Итого: 140



# DATABASES

---

SQLSTATE[08004] [1040] Too many connections



# DATABASES

---

- Масштабируемость
  - Горячие контейнеры
  - Не все SQL решения могут работать с serverless



# DATABASES

---

- Не использовать SQL



# DATABASES

---

- Не использовать SQL
- Открывать и закрывать подключение



# DATABASES

---

- Не использовать SQL
- Открывать и закрывать подключение
- Amazon Aurora Serverless



# DATABASES

---

- Не использовать SQL
- Открывать и закрывать подключение
- Amazon Aurora Serverless
- Не использовать Serverless



# ОТКРЫТЫЕ ВОПРОСЫ

---

- Может нам нужен моно репозиторий?



# ОТКРЫТЫЕ ВОПРОСЫ

---

- Может нам нужен моно репозиторий?
- Как поддерживать версию Lambda?



# ОТКРЫТЫЕ ВОПРОСЫ

---

- Может нам нужен моно репозиторий?
- Как поддерживать версиюность Lambda?
- Нужно ли нам столько Lambda functions?



# ЧТО В ИТОГЕ

---

- Быстрая разработка и проверка теорий



# ЧТО В ИТОГЕ

---

- Быстрая разработка и проверка теорий
- Быстрое масштабирование



# ЧТО В ИТОГЕ

---

- Быстрая разработка и проверка теорий
- Быстрое масштабирование
- Постепенно лучше объединять в сервисы



# ЧТО В ИТОГЕ

---

- Быстрая разработка и проверка теорий
- Быстрое масштабирование
- Постепенно лучше объединять в сервисы
- Логирование в ELK / Внешний сервис



# ЧТО В ИТОГЕ

---

- Быстрая разработка и проверка теорий
- Быстрое масштабирование
- Постепенно лучше объединять в сервисы
- Логирование в ELK / Внешний сервис
- Учитывать особенности внешних приложений



# ЧТО В ИТОГЕ

---

- Быстрая разработка и проверка теорий
- Быстрое масштабирование
- Постепенно лучше объединять в сервисы
- Логирование в ELK / Внешний сервис
- Учитывать особенности внешних приложений
- Serverless framework



---

# ВОПРОСЫ

---



---

# СПАСИБО

Email: [ruslan.ru@gmail.com](mailto:ruslan.ru@gmail.com)

Twitter: [@ruslan\\_firefly](https://twitter.com/ruslan_firefly)

---